ARTÍCULO ORIGINAL / *ORIGINAL ARTICLE*

Data Exporter: A complementary tool to export data simulation from NEURON

Data Exporter: Una herramienta complementaria para exportar datos obtenidos de simulaciones con NEURON

Óscar Emilio Hernández[1], Daladier Jabba[2], Fabián Muñoz[3]

Abstract

Objectives: *To develop a computational tool for NEURON simulation environment, user friendly, to store the results generated during simulation and to make subsequent analysis with other tools such as Matlab and IgorPRo*
Materials and Methods: *Data Exporter was implemented in the programming language hoc. This algorithm is divided in 13 sections or blocks. The first section is necessary to edit in a new simulation. These new configuration determine the geometry, biophysic properties the neurons to simulate and path to save data.*
Results: *To check the efficiency of the algorithm, we simulated the propagation of action potential in TRN(thalamic reticular nucleus) neuron in different large of simulation. We determine that the time of simulation is linear respect to time of simulation.*
Conclusions: *Data Exporter makes easier to start to neural simulation in NEURON reducing the steps of programming to geometry and biophysical properties of the neuron and to allow save data to next steps of analysis.*

Keywords: Data Exporter, Computational simulation tool, NEURON's programming language hoc, Complex neuronal geometry, Action potentials.

Resumen

Objetivos: *Desarrollar una herramienta computacional en ambiente lenguaje de programación hoc de NEURON y de fácil uso, que permita el rápido almacenamiento de los resultados obtenidos para su posterior análisis en otros software tales como Matlab or IgorPro.*

Fecha de recepción: 14 de marzo de 2013
Fecha de aceptación: 5 de julio de 2013

[1] Profesor de tiempo completo, Departamento de Química y Biología, División de Ciencias Básicas, Universidad del Norte.
[2] Profesor de tiempo completo, Departamento de Ingeniería Industrial, División de Ingenierías, Universidad del Norte.
[3] Post-doctoral positions as scientific researcher, Rutgers University, Center for Molecular and Behavioral Neuroscience.

Correspondencia: Óscar Emilio Hernández Bustos, Professor, Departamento de Química y Biología, Universidad del Norte, Km 5, Antigua vía a Puerto Colombia, Barranquilla (Colombia). ohernandezb@uninorte.edu.co, oscaremilio.hernandez@gmail.com.

Salud uninorte
Vol. 29, N° 2, 2013
ISSN 0120-5552

Materiales y métodos: *Para el desarrollo de Data Exporter se escribió un algoritmo en lenguaje de programación hoc de* NEURON. *El algoritmo, escrito en un único archivo de texto, esta dividido en 13 bloques, de los cuales solo el primero debe ser modificado para adaptarlo a una geometría y biofísica neuronal particular y para determinar la ruta de almacenamiento de los datos.*
Resultados: *Se desarrollo un software que simula la propagación de potenciales de acción a través de geometrías neuronales complejas. El uso de esta herramienta permite el almacenamiento de los resultados obtenidos, como potenciales y corrientes de membrana en diferentes puntos de toda la neurona, sin incremento significativo en el tiempo para el desarrollo de los procesos.*
Conclusiones: *Data Exporter es un software que le da mayor flexibilidad a* NEURON *facilitando el acceso a nuevos neurocientíficos, los cuales pueden usarlo con solo conocer los códigos necesarios para el desarrollo de los archivos relacionados con las propiedades geométricas y biofísicas neuronales.*

Palabras clave: Data Exporter, Herramienta de simulación computacional, Lenguaje de programación hoc de NEURON, Geometrías neuronales complejas, Potenciales de acción.

INTRODUCTION

The amount of data collected in neurophysiology experiments for both single neuron and neural networks has increased exponentially during recent years. These data allow the modeling of complex neuronal networks with unprecedented amount of biophysical and anatomical information. These large-scale neural models are frequently non-linear dynamical systems that require numerical simulation to observe their behavior. Currently, there are several specialized software packages available to observe neural phenomena from different perspectives. For example, NEST (1,2) uses single compartmental models; NEURON (3-7) and GENESIS operate with both single and multi-compartmental models (8,9). Other software such as XPPAUT (10,11) is centered primarily on dynamical system analysis.

NEURON is the most popular with a great amount of papers published demonstrating its computationally efficiency in simulations resembling experimental data, especially in problems that range from parts of the single cells to small numbers of cells in which cable properties play a crucial role (12-14). NEURON allows the semiautomatic creation of individual neurons as sections which are subdivided into individual compartments.

The geometry neuron code is based in hoc language, but Python graphic interface is also available. Furthermore, the programs can be written interactively in a command-line shell as well. The properties of the membrane channels of a neuron in terms of kinetic schemes or sets of simultaneous differential and algebraic equation are expressed in NMDOL, a high level programming language (16). Nonetheless, NEURON requires many hours of programming especially when it is necessary to store simulation results for subsequent analysis in other software such as Matlab or IgorPro, for example. Here, we show a complementary code for NEURON 7.0 for Linux version that efficiently exports data without a big impact to the simulation time.

MATERIALS AND METHODS

A simulation in NEURON require, at least, one file with the neurons geometry (.hoc) and other with kinetic properties of each channel (.mod). Simulation conditions are included in the hoc file. In the introduction we mentioned the possibility of building models using graphic interface, however, with the interest to keep short time to simulate, we chose to use hoc language.

Two alternative methods are normally used to build neural geometry with NEURON, Stylized and 3D Method. Stylized specifies the length and diameter of each neuron section (soma, dendrite, etc). 3D Method uses the pt3dadd (x, y, z, diameter) command to define spatial location in Cartesian coordinates and diameter for different neuron (Figure 1A). We chose a 3D Method for this work.

The pt3dadd command is based on using cylindrical or truncated cone shapes (Figure 1B), which it is called line segment. From here, NEURON determines the area to calculate the current density (in mA/cm2) and the membrane potential (in mV) in the center of each cylinder or truncated cone. These geometric parameters should be stored previously to any posterior simulation (17,18).

The simulation were tested on a standard PC (an Intel ® Core ™2 Duo processor of 2.66Hz and 3.468GB of RAM memory).

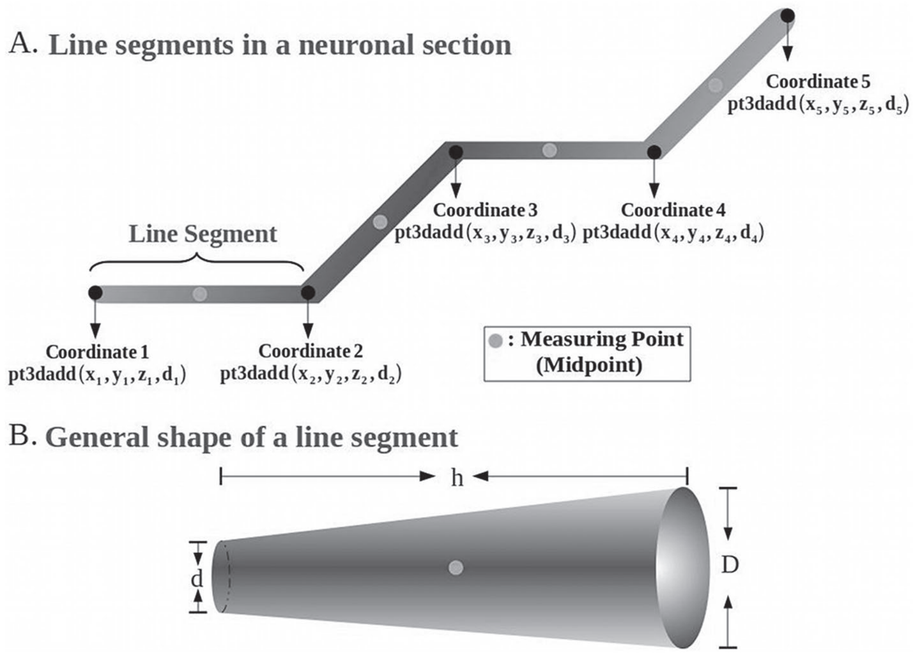

A. Line segments in a neuronal section
Line Segment
Coordinate 1
pt3dadd(x_1, y_1, z_1, d_1)
Coordinate 2
pt3dadd(x_2, y_2, z_2, d_2)
Coordinate 3
pt3dadd(x_3, y_3, z_3, d_3)
Coordinate 4
pt3dadd(x_4, y_4, z_4, d_4)
Coordinate 5
pt3dadd(x_5, y_5, z_5, d_5)
●: Measuring Point (Midpoint)
B. General shape of a line segment
h
d
D

Figure 1. Neural structure generate with NEURON. A. General view of neural section. The straight line between two successive coordinates is defined according to command pt3dadd (x, y, z, diam). The central colored dot of each segment is the point where are measured the parameters (voltage, current density). B) Details of one section and the parameters that define it.

Data Exporter is a routine that stores current and voltage data from simulation in .dat files (Figure 2). Additionally, two more text files are generated. The first file (neuronal_geometry.txt) is neuron morphology, it describes the Cartesian coordinates (x, y, z) and diameters for each line segment.

The second (general_data_relevant.txt) has general information related to simulation conditions, for example, total time (ms), stimulus frequency (Hz), number of line segments, neural sections, files generated with membrane currents and membrane current densities and approximation of total superficial area of neuron membrane (μm^2).

RESULTS

Tools Details

Data Exporter will generate a folder called "OUTPUT" in the username main folder where data will be stored (Figure 3 and 4). Three sub-folders will also be included: CURRENT_LS (membrane currents), CURRENT_DENSITIES_MPLS (membrane current densities) and POTENTIALS_MPLS (membrane potentials). The term MPLS refers to Membrane Potential by Lineal Segment.

We designed a main script, DATA_EXPORTER.hoc, divided into 13 comment blocks, with a series of modifiable fields that assign software conditions to a specific neuron type. The meaning can be checked inside each section (Figure 2). These fields can be modified before a new simulation in necessary case.

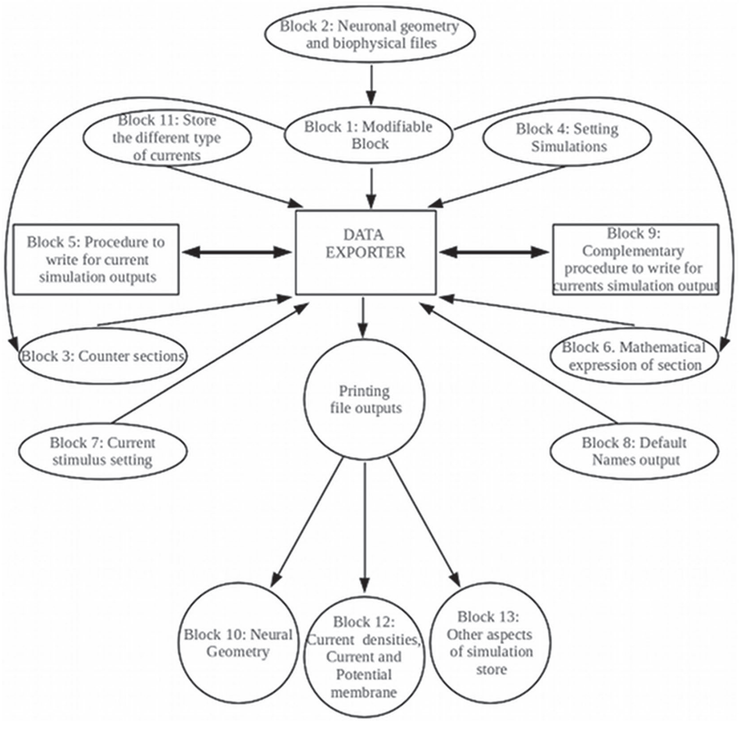

Block 2: Neuronal geometry and biophysical files
Block 11: Store the different type of currents
Block 1: Modifiable Block
Block 4: Setting Simulations
Block 5: Procedure to write for current simulation outputs
DATA EXPORTER
Block 9: Complementary procedure to write for currents simulation output
Block 3: Counter sections
Block 6. Mathematical expression of section
Printing file outputs
Block 7: Current stimulus setting
Block 8: Default Names output
Block 10: Neural Geometry
Block 12: Current densities, Current and Potential membrane
Block 13: Other aspects of simulation store

Figure 2: Data Exporter structure. Each block name and dependency is presented in diagram to explain the code in Data Exporter, details are in the text.

***Block 1:** Username*

First, it should be modified the field user_name by corresponding name of user.

user_name = "user_name"

Next, it should be included the hoc files name that defined the geometry and biophysical characteristics of a neuron in following fields:

Geometry = "geometry_file.hoc"

Biophysics ="biophysics_file.hoc"

Now, the names of currents involved in the neuron (ionics, passive, and capacitive) are added. For example,

list_curr.o(0).s = "ina"(sodium current)

list_curr.o(1).s = "ik"(potassium current)

list_curr.o(2).s = "i_pas" (passive current)

list_curr.o(3).s = "i_cap"(capacitive current)

list_curr.o(4).s = "ica"(calcium current)

To add more currents in case were necessary, follow this syntax list_curr.o (i).s = "current_i", where "i" corresponds to the name of the new current. If there are less currents that defined by default only delete from the list. Subsequently, you can set the position where the current will be measured in the section. In NEURON, every section has length equal 1, then electrode could located in coordinates goes between 0 to 1, in this example we chose at the middle of the section.

section_stim = "section_name"

x_stim = 0.5 (middle section)

The last list of parameters correspond to simulation conditions as:

Minimum time interval or time lapse (ms): st = 0.01

Total simulation time (in ms): tts = 5

Temperature (ºC): tgc= 40

Number of current stimulus (square pulses): ns = 4

Time between stimulus current pulses (ms): ipt = 2

Time delay of stimulus initiation (ms): sit = 10

Scope of each current stimulus (nA): ampl = 5

Duration of each current stimulus (ms): durat = 30

The last parameter is the number of dat output files. We fixed to generate up to 50000 , which is written as nma_dat = 50000

***Block 2:** Imported geometry and Biophysical Files*

Here, the geometry biophysics files are imported explicitly using xopen( ) function. After that is created the complete list of sections extracted from imported files.

***Block 3:** Counter Sections*

A counter was defined to include the total number of section created (dendrites, soma and axon). This parameter is possible to display in the NEURON terminal prompted writing vec_sum_segl.sum( ).

***Block 4:** Setting Simulations*

The function param_initio( ) load the simulation conditions set up in block 1.

***Block 5**: Procedure to write output currents simulation - Part 1*

The block creates the procedure to save the output currents of the simulation and these outputs are stored in definition_vector_register.hoc file.

Block 6: *Mathematical expression of Section*

We defined the segment area under the idea that each section is a truncate cone, as follows:

$$A = \frac{\pi}{4}(D+d)\sqrt{(D-d)^2+4h^2} \quad (1)$$

where D and d are the major and minor diameters of the line segment (Figure 1), this is applied to every section in block 9, which were fixed in block 3.

Block 7: *Current stimulus setting*

This block create the external current pulses (time start, duration, amplitude and frequency) that were set on block1.

Block 8: *Default names output*

The file names of currents, currents densities and potentials are establish as membrane_currents_xxx.dat, membrane_currents_densities_xxx.dat and membrane_potential_xxx.dat, where xxx are signed the time when was stored.

Block 9: *Procedure to write output currents simulation - Part 2*

Complementary to block 6. These code lines use the equation in block 6 adapted to each segment (axon, dendrites, soma) to measure the current densities, currents and membrane After that, they are stored in the output file (variable_vector_register.hoc).

Block 10: *Neural geometry*

The neuronal anatomy is stored in file txt (neuronal_geometry.txt). In contrast, geometry.hoc that we mention above, here is explicit every component (xyz coordinates, name section) to difference geometry.hoc that only are the coordinate and general expression about spatial feature of sections. The data is ordered in columns: a cardinal number, section, (x,y,z) coordinates initial section, diameter initial section, (x,y,z) coordinates end section, diameter end section.

Block 11:

Here is the procedure that creates the vector that store the different type of currents from the simulation.

Block 12: *Printing File outputs*

This section is a code to print variables created during simulation in the output files (CURRENTS_DENSITIES_MPLS, CURRENT_LS, POTENTIAL_MPLS)

Block 13:

Finally, others aspects of simulation are stored general_data_relevant.txt, for example: total time, stimulus frequency, and number of sections. And some statistics as the number of densities, currents and potentials files is created along with the lateral area neuron.

DISCUSSION

Performance tool

In order to show tool functionality, Thalamic Reticular Nucleus neuron (TRNn) (Figure 4D) was used as a test. The neuron geometry and biophysics (Table 1) data were obtained from the ModelDB (http:// senselab.med.yale. edu/ModelDB/defa ult. asp), a NEURON database available. The simulation started with a square current pulse as shown in Figure 4C in the soma the TRNn and density current and membrane potential in a dendrite were registered (Figure 4A, 4B).

In Figure 5, we show the output results from Data Exporter, as it was described in block 8 (Tools Details). In detail, a dendrite was labeled as dend1 [6] according to NEURON language (Figure 4B). The spatial neuron coordinates were stored in the neuronal_geometry.txt. To continue, the folders CURRENT_DENSITIES_MPLS and POTENTIAL_MPLS .dat files are labeled in relation to time simulation (Figure 5). When these files are opened we see two columns, the first corresponds to line segment and the second to membrane current densities or membrane potentials, with this information is useful to analysis with other pieces of software.

In terms of time processing, the implementation showed a linear relation between time simulation and number of files were printed (Table 2).

Table 1. Inserted mechanisms and maximum conductance values in model TRNn. HH: Maximum conductance on Hodgkin and Huxley style (Sodium: gNa, Max. and Potassium: gK, Max.), Pas: Maximum passive conductance (gpas, Max.), It: Maximum calcium conductance (gCa, Max.).

Neuron section	Inserted mechanisms	gNa,Max. (S/cm2)	gK,Max. (S/cm2)	gCa,Max. (S/cm2)	gpas,Max. (S/cm2)
Soma	HH, Pas, It	0.5	0.1	8x10-4	5x10-4
Dendrites		0	0	2x10-4	5x10-4

Source: Tabulated by authors

Table 2. General results of action potentials propagation simulation through TRNn. 5 different times were simulated (first column), with different text files obtained for each case (third column), and different real simulation times (second column)

Simulated time (ms)	Real time (min)	Nº .dat files	dt (ms)	Nº Line segments	Nº Sections
200	8,0	60000	0.01	1444	81
150	6,0	45000	0.01	1444	81
100	4.2	30000	0.01	1444	81
50	2.1	15000	0.01	1444	81
10	0.1	3000	0.01	1444	81

Source: Tabulated by authors

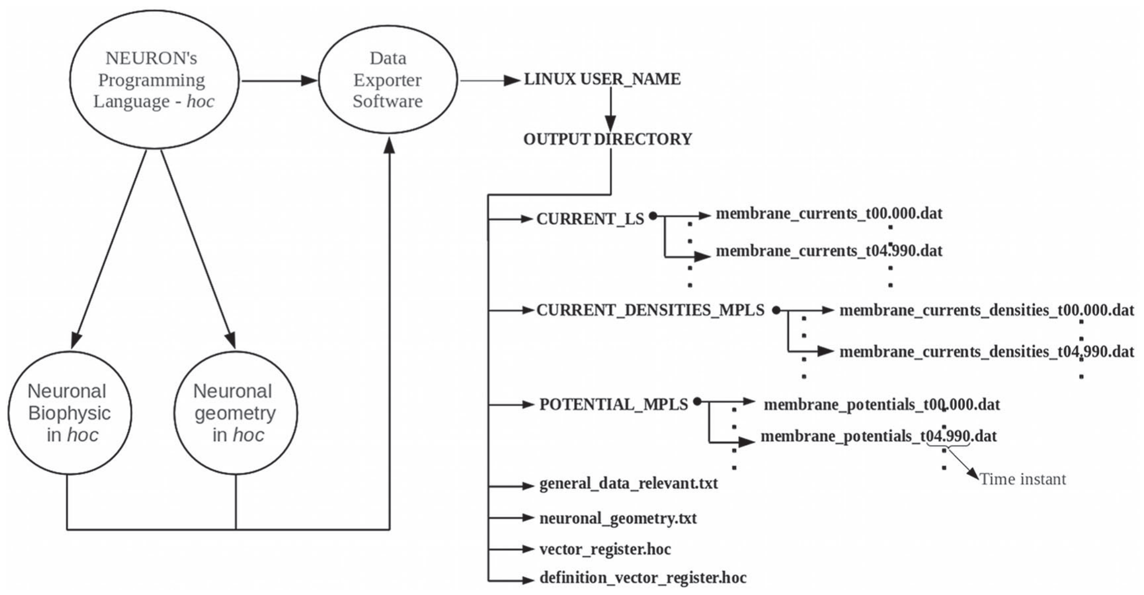

NEURON's Programming Language - *hoc*
Data Exporter Software
LINUX USER_NAME
OUTPUT DIRECTORY
CURRENT_LS
membrane_currents_t00.000.dat
membrane_currents_t04.990.dat
CURRENT_DENSITIES_MPLS
membrane_currents_densities_t00.000.dat
membrane_currents_densities_t04.990.dat
POTENTIAL_MPLS
membrane_potentials_t00.000.dat
membrane_potentials_t04.990.dat
Time instant
general_data_relevant.txt
neuronal_geometry.txt
vector_register.hoc
definition_vector_register.hoc
Neuronal Biophysic in *hoc*
Neuronal geometry in *hoc*

Figure 3: General structure of one cycle of simulation. Here, it is exposed in details, the outputs (CURRENT_LS, CURRENT, DENSITIES_PMLS and POTENTIAL_MPLS) and the file names are created from Data Exported.

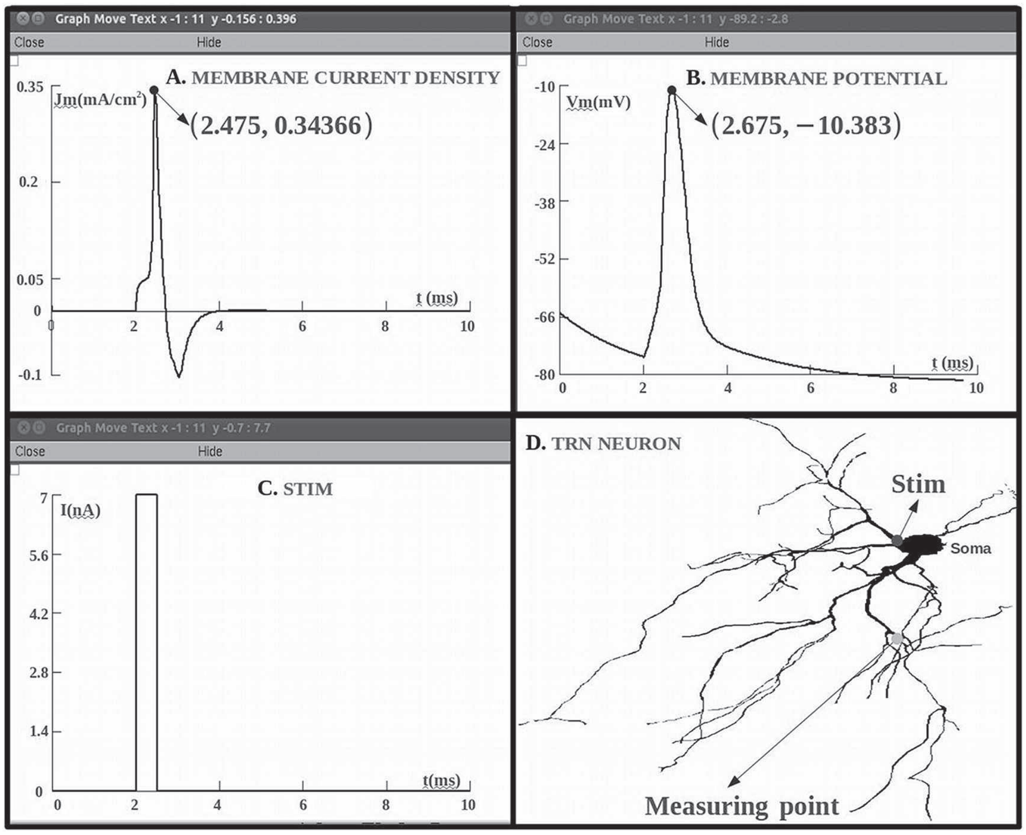
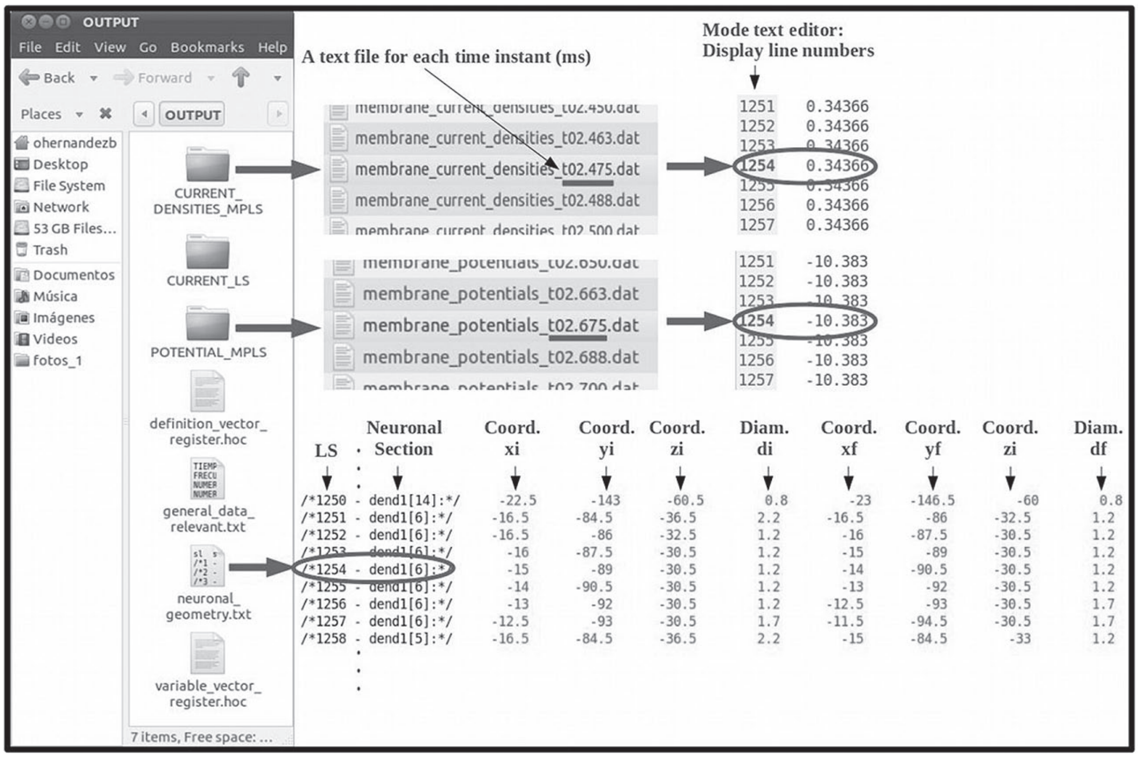

Graph Move Text x -1 : 11 y -0.156 : 0.396
Close
Hide
A. MEMBRANE CURRENT DENSITY
Jm(mA/cm²)
(2.475, 0.34366)
t (ms)
Graph Move Text x -1 : 11 y -89.2 : -2.8
B. MEMBRANE POTENTIAL
Vm(mV)
(2.675, −10.383)
t (ms)
Graph Move Text x -1 : 11 y -0.7 : 7.7
C. STIM
I(nA)
t(ms)
D. TRN NEURON
Stim
Soma
Measuring point

Figure 4: Graphics generated through the NEURON graphical interface. A) Current membrane density. B) Membrane potential. C) Stimulus and D) Reticular Thalamus neuron anatomy. Note that in A and B, current and potential density values and the corresponding time instants.



```
membrane_current_densities_t02.450.dat
membrane_current_densities_t02.463.dat
membrane_current_densities_t02.475.dat
membrane_current_densities_t02.488.dat
membrane_current_densities_t02.500.dat
```

```
1251    0.34366
1252    0.34366
1253    0.34366
1254    0.34366
1255    0.34366
1256    0.34366
1257    0.34366
```

```
membrane_potentials_t02.650.dat
membrane_potentials_t02.663.dat
membrane_potentials_t02.675.dat
membrane_potentials_t02.688.dat
membrane_potentials_t02.700.dat
```

```
1251    -10.383
1252    -10.383
1253    -10.383
1254    -10.383
1255    -10.383
1256    -10.383
1257    -10.383
```

LS	Neuronal Section	Coord. xi	Coord. yi	Coord. zi	Diam. di	Coord. xf	Coord. yf	Coord. zi	Diam. df
/*1250 -	dend1[14]:*/	-22.5	-143	-60.5	0.8	-23	-146.5	-60	0.8
/*1251 -	dend1[6]:*/	-16.5	-84.5	-36.5	2.2	-16.5	-86	-32.5	1.2
/*1252 -	dend1[6]:*/	-16.5	-86	-32.5	1.2	-16	-87.5	-30.5	1.2
/*1253 -	dend1[6]:*/	-16	-87.5	-30.5	1.2	-15	-89	-30.5	1.2
/*1254 -	dend1[6]:*/	-15	-89	-30.5	1.2	-14	-90.5	-30.5	1.2
/*1255 -	dend1[6]:*/	-14	-90.5	-30.5	1.2	-13	-92	-30.5	1.2
/*1256 -	dend1[6]:*/	-13	-92	-30.5	1.2	-12.5	-93	-30.5	1.7
/*1257 -	dend1[6]:*/	-12.5	-93	-30.5	1.7	-11.5	-94.5	-30.5	1.7
/*1258 -	dend1[5]:*/	-16.5	-84.5	-36.5	2.2	-15	-84.5	-33	1.2

Figure 5: Results scheme from Data Exporter. This shows how the specific value of a variable in a given time instant should be located.

In conclusion, Data Exporter makes easier to start to neural simulation in NEURON reducing the steps of programming to geometry and biophysical properties of the neuron.

In the future, we expect to improve this tool by including a graphical interface compatible with the NEURON as well as exporting other variables such as extracellular potentials. Also adding support for Windows and Mac OS.

Financial Disclosure

This project was financed by Universidad del Norte.

REFERENCES

(1) Gewaltig M, Diesmann M. NEST. *Scholarpedia* 2007; 2(4): 1430

(2) NEST Initiative. Retrieved: 27 June 2013. Available at: http://www.nest-initiative.org/index.php/Software:About_NEST 2

(3) Hines M. A program for simulation of nerve equations with branching geometries. *International Journal of Bio-Medical Computing* 1989; 24: 55-68.

(4) Hines M. NEURON - A Program for Simulation of Nerve Equations. *Kluwer Academic Publishers* 1993; pp. 127-136.

(5) Hines M, Carnevale N. The NEURON Simulation Environment. *Cambridge: MIT Press* 1993. pp. 127-136.

(6) NEURON. Retrieved: 27 June 2013. Available at: http://www.neuron.yale.edu/neuron/

(7) Hines M, Carnevale N. Discrete event simulation in the NEURON environment. *Neurocomputing in press* 2004; 58-60: 117-1122.

(8) Bower D. The Book of GENESIS: Exploring Realistic Neural Models with GEneral NEural SImulation System. *Springer-Verlag* 1998.

(9) GENESIS. Retrieved: 27 June 2013. Available at:http://www.genesissim.org/GENESIS/

(10) Ermentrout B. Simulating, analyzing, and animating dynamical systems: a guide to XPPAUT for researchers and students. *Society for Industrial and Applied mathematics (SIAM Review)* 2002.

(11) XPPAUT. Retrieved: 27 June 2013. Available at: http://www.math.pitt.edu/~bard/xpp/xpp.html

(12) Destexhe T. Contreras D.and Sejnowski A. Steriade A. Model of spindle rhythmicity in the isolated thalamic reticular nucleus. *Journal of Neurophysiology* 1989; 72: 803-818.

(13) Destexhe A, Contreras D, Steriade A, Sejnowski T, Huguenard J. In vivo, in vitro, and computational analysis of dendritic calium currents in thalamic reticular neurons. *Journal of Neuroscience* 1996; 16: 169-185.

(14) NEURON Models-ModelDB. Retrieved: 27 June 2013. Available at: http://senselab.med.yale.edu/modeldb/

(15) King J, Hines M, Hill S, Goodman P, Markram H, Schürmann F. A component based extension framework for large-scale parallel simulations in neuron. *Frontiers in Neuroinformatics* 2009; 3: 1-11.

(15) Hines M. Expanding neuron's repertoire of mechanisms with nmodl. *Neural Computation* 2000;12: 839-851.

(17) Koch C, Gold C, Darrel A, Buzaki G. On the origin of the extracellular action potential waveform: A modeling study. *Journal of Neurophysiology* 2006; 95(5): 311-328.

(18) Holt G, Koch C. Electrical Interactions Via the Extracellular Potential Near Cell Body. *Journal of Computational Neuroscience* 1999; 6(2): 169-184.